Refinement and Verification of Synchronized Component-based Systems

Olga Kouchnarenko, Arnaud Lanoix

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Refinement and Verification of Synchronized Component-based Systems

Olga Kouchnarenko — Arnaud Lanoix

N° 4862

Juin 2003

THÈME 2

ISSN 0249-6399 ISRN INRIA/RR--4862--FR+ENG

INRIA
LORRAINE

Refinement and Verification of Synchronized Component-based Systems

Olga Kouchnarenko* , Arnaud Lanoix†

Thème 2 — Génie logiciel
et calcul symbolique
Projet CASSIS

Rapport de recherche n° 4862 — Juin 2003 — 29 pages

Abstract: This article deals with specification, refinement and verification approaches for systems designed with synchronized components. First of all, we define a synchronized composition of components. Transition systems are used to specify or/and to model synchronized component-based systems. Second, we give refinement semantics for these component-based systems before proposing a method to verify the refinement of a whole system from the weak refinement of its components. We also present *SynCo* (for Synchronized Component-based Systems): a tool we are implementing using our method. Third, a compositional way to verify safety properties is proposed: the unreachability of a (set of) state(s) can be efficiently ensured for a synchronized component-based system.

The different aspects of our work are illustrated on an industrial example of a wind-screen wipers system composed of a control lever, a rain sensor and two (left and right) wind-screen wipers.

Key-words: composition, synchronization, compositional verification, refinement, reachability analysis

* LIFC – Université de Franche-Comté, email: kouchna@univ-fcomte.fr
† LIFC – Université de Franche-Comté, email: lanoix@univ-fcomte.fr

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy (France)
Téléphone : +33 3 83 59 30 00 — Télécopie : +33 3 83 27 83 19

Raffinement et vérification de systèmes à composants synchronisés

Résumé : Cet article traite de la spécification, du raffinement et de la vérification de systèmes conçus sous forme de composants synchronisés. Pour spécifier et/ou pour modéliser ces systèmes, nous utilisons des systèmes de transitions. Nous définissons d'abord une composition synchronisée de systèmes et donnons la sémantique du raffinement pour ce type de systèmes. Ensuite nous donnons des conditions nécessaires et suffisantes du raffinement du système complet à partir du raffinement affaibli de ses composants. Un algorithme de vérification du raffinement pour ce type de système est implanté dans *SynCo* (pour Synchronized Component-based Systems) Finalement, un algorithme de vérification compositionnelle de propriétés de sûreté est proposé.

Les différents aspects de ce travail sont illustrés par l'exemple industriel d'un controleur d'essuyage composé d'un levier de controle, d'un capteur de pluie et de deux (gauche et droite) essuie-glaces.

Mots-clés : composition, synchronisation, vérification compositionnelle, raffinement, vérification algorithmique, analyse d'atteignabilité

1 Introduction

Verification across abstraction and refinement steps is a central and important issue in formal system validation. It presents both practical and theoretical difficulties that have not yet been satisfactory solved.

In this paper, we deal with specification and verification of component-based finite state systems supporting a top-down refinement paradigm. We suppose a specification obtained by a refinement process. Several methods, like B [1, 4], $TLA+$ [20], $CSP2B$ [12], etc. propose a refinement based development. The system specification and modelling we consider in this paper, are inspired by the syntactic and semantic concepts of the B refinement method which has been successfully used to specify many reactive systems. On overview, the reader can refer to case studies such as an industrial automatism [2], as well as industrial applications such as *MÉTÉOR* [5] by *Matra Transport International*, and the *SPECTRUM* project [28] by *GEC-Marconi Avionics Limited*.

In [16, 18], it has been proposed to enrich B specifications with dynamic properties formulated in the Propositional Linear Temporal Logic ($PLTL$). In [8], we express the refinement semantics as a relation between transition systems. In general, behavioural properties established for an abstract system model are not preserved when the system is refined to a richer level of detail. It is not the case for us: in [14], we show that our refinement relation preserves the abstract system $PLTL$ properties. This way, an algorithmic verification of the refinement by model exploration can be associated with the verification of the $PLTL$ properties, by model-checking.

However, it is well-known that the algorithmic verification quickly meets its limits when applied to huge systems. Therefore, we have to face the problem of combinatorial explosion during refinement since details introduced by the refinement tend to drastically increase the number of states of the systems. It is also the case while verifying properties by model-checking. Compositional approaches partially avoid this problem.

A way to have components in the B method is to decompose the application into separate machines. There are many works on structured development using decomposition into machines and refinement (see, for example [13, 11, 27]). Generaly, B event systems which are closed systems can be used to describe the components. So, the interactions between components have to be described independently. This is the idea used for example in [12] or in [29].

Unlike the Schneider and Treharne's approach [26], we propose in [9] to remain in the framework of the B event systems. We assume that components do not share variables and we propose to specify a synchronization as pairs of events belonging

to two different components with feasibility conditions. These conditions achieve the synchronization by constraining the activation of events of a component by a predicate over the variables of another component. Moreover, during the refinement verification we take the external non-determinism into account.

In this paper, we go further to conciliate the synchronized component-based specification with the refinement verification. First, we propose to use transition systems both for specifying and for modelling synchronized component-based systems. Second, we define a refinement relation of component-based systems which is weaker than the refinement relation defined in [9]. Third, we give the conditions to ensure the refinement of the whole system from the weak refinements of its synchronized components, and vice versa. Furthermore, we propose to exploit the refinement to verify safety properties in a compositional way. The interest of this proposition is that the component-based refinement guarantees preservation of the abstract systems properties for the refined systems.

As a comparison, our goal for addressing the B event systems composition is different from the B *decomposition* one proposed by J.-R. Abrial. In [3], the event systems decomposition is based on extern shared variables and synchronized events. The main advantage of our technique is that it frees the user from the whole system design.

This paper is organised as follows. After giving preliminary notions, we define, in Section 2, the behavioural semantics of synchronized component-based systems. In Section 3, we define a refinement relation for these systems and explain our main compositionality result using for the compositional refinement verification. Then, in Section 4, we investigate how our approach is used in the context of a compositional safety property verification. Section 5 introduces a tool implementing the synchronized component refinement verification. Throughout this paper, we illustrate the use of our framework on an industrial example of a wind-screen wipers system. We end by some perspectives.

2 Synchronized Parallel Composition

In this section, we introduce interpreted labelled transition systems to specify and to model the behaviours of a component. We specify the synchronized behaviours of components by tuples of labels with feasibility conditions constraining activations of transitions with these labels. We provide a definition for a context-in component (i.e. a component in the context of the others). This definition is required to define the synchronized parallel composition of components under a synchronization.

Let $Var = \{X_1, \ldots, X_n\}$ be a finite set of variables with their respective domains $\mathbb{D}_1, \ldots, \mathbb{D}_n$. Let AP be a set of atomic propositions $ap \stackrel{\text{def}}{=} (X_i = v)$ with $X_i \in Var$ and $v \in \mathbb{D}_i$. Let SP be a set of state propositions sp defined by following grammar: $sp_1, sp_2 ::= ap \mid \neg sp_1 \mid sp_1 \vee sp_2$.

Definition 1 (Interpreted Labelled Transition System (LTS))
A interpreted labelled transition system S over Var is a tuple $< Q, Q_0, E, T, l >$ where:
- *Q is a set of states,*
- *$Q_0 \subseteq Q$ is a set of initial states,*
- *E is a finite set of transitions labels or actions,*
- *$T \subseteq Q \times E \times Q$ is a labelled transition relation, and*
- *$l : Q \rightarrow SP$ is an interpretation of each state on the system variables.*

We define the sum of two transition systems over the same set of variables Var. Furthermore, we will show how to compute a parallel composition of synchronized components using this operator.

Definition 2 (Sum of Two LTSs)
Let $S_1 =< Q_1, Q_{01}, E_1, T_1, l_1 >$ and $S_2 =< Q_2, Q_{02}, E_2, T_2, l_2 >$ be two transition systems over Var. The sum of S_1 and S_2, written $S_1 \uplus S_2$, is $< Q_1 \cup Q_2, Q_{01} \cup Q_{02}, E_1 \cup E_2, T_1 \cup T_2, l_{12} >$ where l_{12} is defined by:

$$l_{12}(q) = \begin{cases} l_1(q) \ if \ q \in Q_1, \\ l_2(q) \ if \ q \in Q_2, \end{cases}$$

Moreover, $\forall q_1.\ q_1 \in Q_1,\ \forall q_2.\ q_2 \in Q_2.\ (l_1(q_1) = l_2(q_2) \Leftrightarrow q_1 = q_2)$.

In order to define our synchronized parallel composition, we give the definition of a *synchronization* of n components. For that, we introduced a new transition label '$-$' for the fictive action "skip".

Definition 3 (Synchronization of n Components)
*Let $S_1, \ldots, S_n$ be n components. A synchronization Synch is a set of elements (α **when** p) where:*
- *$\alpha = (e_1, \ldots, e_n) \in \prod_{i=1}^{n} (E_i \cup \{-\})$,*
- *p is a state proposition on the components variables.*

To illustrate the previous definitions, we introduce the example of a car windscreen wipers system. Consider the wipers system WS_A composed by a control lever,

a rain sensor and two (left and right) wind-screen wipers. The control lever CO_A can select the mode of the wiper system: *co=manual*, *co=auto* or *co=stop*. Its behaviour is shown in Fig. 1. The left and the right wipers have the same behaviour. The transition system in Fig. 2 shows two positions for the left wiper LW_A: *lw=leftUp* or *lw=leftDown*. It is the same thing for the right wiper RW_A. The rain sensor SE_A can detect the rain amount (*se=not*, *se=tiny* or *se=strong*). Moreover, it can be off (*se=off*) (see Fig. 3).

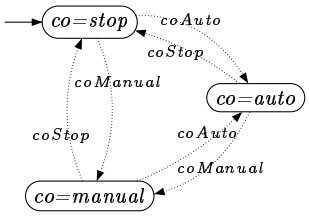

Figure 1: CO_A

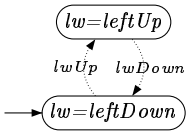

Figure 2: LW_A

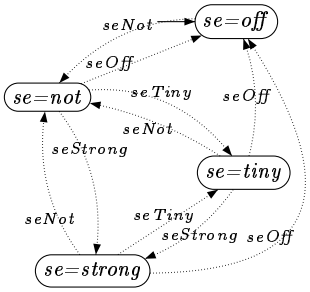

Figure 3: SE_A

(lwDown,rwDown) **when** *((co=manual) ∧ (se=off)) ∨ ((co=auto) ∧ (se=tiny)) ∨ ((co=auto) ∧ (se=strong))*,
(lwUp,rwUp) **when** *((co=manual) ∧ (se=off)) ∨ ((co=auto) ∧ (se=tiny)) ∨ ((co=auto) ∧ (se=strong))*,
coManual **when** *(lw=leftDown) ∧ (rw=rightDown) ∧ (co=stop) ∧ (se=off)*,
coStop **when** *(lw=leftDown) ∧ (rw=rightDown)∧ (co=manual) ∧ (se=off)*,
seTiny **when** *(co=auto) ∧ (lw=leftDown) ∧ (rw=rightDown) ∧ (se!=off)*,
seStrong **when** *(co=auto) ∧ (lw=leftDown) ∧ (rw=rightDown) ∧ (se!=off)*,
seNot **when** *(co=auto) ∧ (lw=leftDown) ∧ (rw=rightDown) ∧ (se!=off)*,
(coManual, seOff) **when** *(co=auto) ∧ (lw=leftDown) ∧ (rw=rightDown)*,
(coStop,seOff) **when** *(co=auto) ∧ (lw=leftDown) ∧ (rw=rightDown)*,
(coAuto,seNot) **when** *(se=off) ∧ (lw=leftDown) ∧ (rw=rightDown)*

Figure 4: Synchronization csw_A

A synchronization csw_A is given to describe the authorized behaviours of the complete system WS_A. For example, the left and right wipers must move together ((*lwDown,rwDown*) and (*lwUp,rwUp*)), the rain sensor sends information only if the control lever mode is *auto* (*seTiny*, *(seStrong* and *seNot*), the control lever mode or the rain sensor can change only if the wipers are down ((*coManual,seOff*), (*coAuto,seNot*)), etc. csw_A is given in Fig. 4. For readability reason the fictive transition label '-' is not shown in csw_A.

Each component $S_1, \ldots, S_n$ is a context-free component. However, to take synchronization into account, we need to define a context-in component, i.e. a component in the context of the others under a synchronization *Synch*.

Definition 4 (Context-in Component)
Let $S_1, \ldots, S_n$ be n components. Let Synch be their synchronization. A context-in component S_i^c is defined by the tuple $< Q_i^c, Q_{0i}^c, E_i^c, T_i^c, l_i^c >$ where:
- $Q_i^c \subseteq Q_1 \times \ldots \times Q_n$ *with* $(q_1, \ldots, q_n) \in Q_i^c$,
- $Q_{0i}^c \subseteq Q_{01} \times \ldots \times Q_{0n}$,
- $E_i^c = \{(e_1, \ldots, e_i, \ldots, e_n) \mid (((e_1, \ldots, e_i, \ldots, e_n)$ **when** $p) \in Synch)$ $\wedge (e_i \in E_i)\}$,
- $l_i^c((q_1, \ldots, q_n)) = l_1(q_1) \wedge \ldots \wedge l_n(q_n)$,
- $T_i^c \subseteq Q_i^c \times E_i^c \times Q_i^c$ *with* $((q_1, \ldots, q_n), (e_1, \ldots, e_n), (q'_1, \ldots, q'_n)) \in T_i^c$ *iff:*
  - $((e_1, \ldots, e_n)$ **when** $p) \in Synch$,
  - $l_i^c((q_1, \ldots, q_n)) \Rightarrow p$, *and*
  - $\forall k.(k \in \{1, \ldots, n\} \Rightarrow ((e_k = - \wedge q_k = q'_k) \vee (e_k \neq - \wedge (q_k, e_k, q'_k) \in T_k)))$.

All context-in components have the same set of variables $Var = \bigcup_{j=1}^{n} Var_j$. Note that the graph representing a context-in component may be unconnected.

Figure 5 shows a representation of the context-in component SE_A^c that presents all behaviours of SE_A under csw_A. It is either simple behaviours (*seNot*, *seTiny*, *seStrong*) or synchonized behaviours ((*seOff,coManual*), (*seNot,coAuto*)). We have similar figures for the context-in components CO_A^c, LW_A^c and RW_A^c. Remark

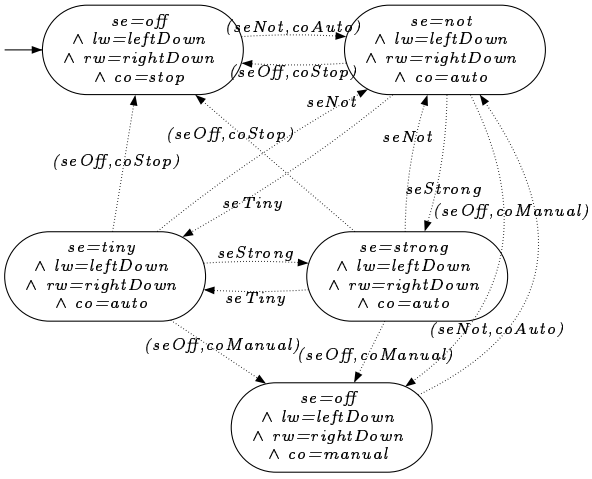

Figure 5: SE_A^c

that RW_A^c and LW_A^c are identical. Indeed, all behaviours of LW_A and RW_A are synchronized ((*lwDown,rwDown*) and (*lwUp,rwUp*)).

The whole system is a rearrangement of its separate parts, i.e., the components and their interactions. This arrangement is specified by a synchronized composition between components. We define a synchronized composition of n components $S_1, \ldots, S_n$ under a synchronization *Synch* by the sum of the n context-in components $S_1^c, \ldots, S_n^c$.

Definition 5 (Synchronized Composition of n Components)
Let $S_1, \ldots, S_n$ be n components and Synch their synchronization. Let $S_1^c, \ldots, S_n^c$ be their respective context-in components. The synchronized parallel composition of $S_1, \ldots, S_n$ under Synch is defined by:

$$\|_{Synch}(S_1, \ldots, S_n) \stackrel{def}{=} \biguplus_{i=1}^{n}(S_i^c)$$

Notice that our parallel composition is more expressive than the classical synchronized product of transition systems. This is due to feasibility conditions, i.e., predicates over variables of the n components that constrain the composition. It is not a designer's task to write the context-in components nor their sum. As they can be automatically generated (thanks to Definitions 2 and 4), the designer only specifies the components and the synchronization.

Recall that the four context-in components have the same set of variables $Var_A = \{co, lw, rw, se\}$. The wipers system $WS_A = \|_{csw_A}(CO_A, SE_A, LW_A, RW_A)$ equals to $CO_A^c \uplus SE_A^c \uplus LW_A^c \uplus RW_A^c$.

3 Synchonized Component-based Systems Refinement

In this section, we first recall the refinement semantics given in [8, 9] for classical one component transition systems. A refinement relation in the style of the Milner-Park simulation relation between abstract and concrete transition systems is given. Second, we want these semantics to fit with the synchronized component-based systems. For that we define a refinement relation between transition systems which is weaker than the refinement relation introduced in [8, 9]. Nevertheless, this weak refinement allows us to ensure the refinement of the whole system in a compositional way, unlike the refinement defined in [8, 9]. We begin by an example before giving formal definitions.

Consider a refinement of the wind-screen wipers system, still composed of four components: CO_R, SE_R, LW_R and RW_R. In CO_R the speed choice *seCo* is per-

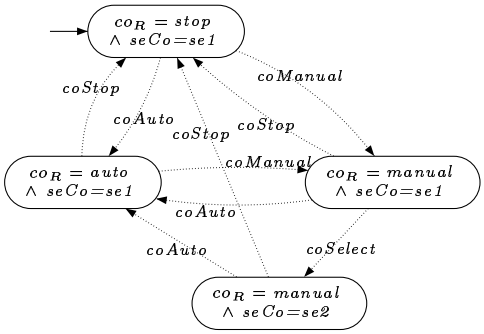

Figure 6: CO_R

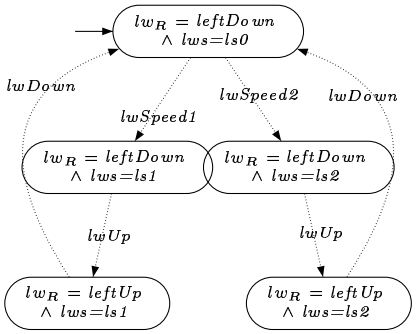

Figure 7: LW_R

formed in the manual mode: *seCo=se1* for speed 1 or *seCo=se2* for speed 2 (see Fig. 6). In both LW_R and RW_R the wind-screen wipers speed is taken into account. For LW_R a new variable *lws* (for left wiper speed) is introduced with $\mathbb{D}(lws) = \{ls0, ls1, ls2\}$ for respectively, no speed, speed 1 or speed 2 (see Fig. 7). Idem for RW_A. In the abstract system SE_A, there are two rain levels. We observe in SE_R both limits of the tiny-rain level (se_R=*maxTiny*) and the strong-rain level (se_R=*minStrong*).

The authorized behaviours of the refined wipers system WS_R are described in the synchronization csw_R. We still have the old synchronized behaviours, i.e. left and right wipers must move together ((*lwDown,rwDown*) and ((*lwUp,rwUp*)), the control lever and the rain sensor change only if the wipers are down (*coManual*, *coStop*, *seTiny*, *seStrong*), etc. In addition, some of the new transitions are also synchronized. For example, the speed selection in the control lever manual mode can happen only if the wipers are down (*coSelect*). The wipers change their speeds simultaneously ((*lwSpeed1,rwSpeed1*) and (*lwSpeed2,rwSpeed2*)). In the control lever auto mode, the tiny-rain detection is synchronized with the level 1 of the wipers speeds ((*seMaxTiny,lwSpeed1,rwSpeed1*)), and the strong-rain detection is synchronized with the level 2 of the wipers speeds ((*seMinStrong,lwSpeed2,rwSpeed2*)).

3.1 Basic Transition Systems Refinement

Let $SA =< Q_A, Q_{0A}, E_A, T_A, l_A >$ be an abstract transition system over Var_A and $SR =< Q_R, Q_{0R}, E_R, T_R, l_R >$ a concrete transition system over Var_R. In this section, some basic definitions about transition systems refinement are given. The syntactic concepts of the refinement are the following. Refinement introduces new

transition labels, so $E_A \subseteq E_R$. Refinement introduces new variables and renames abstract ones, so $Var_A \cap Var_R = \varnothing$.

Let GI be a formula over $SP_A \cup SP_R \cup SP'$ where SP_A is over Var_A, SP_R is over Var_R, and $SP' \stackrel{\text{def}}{=} \{X_A = X_R\}$ with $X_A \in Var_A$ and $X_R \in Var_R$ and $\mathbb{D}(X_A) = \mathbb{D}(X_R)$. GI is commonly known as a gluing invariant linking variables of the abstract and concrete systems. A binary relation $\mu \subseteq Q_R \times Q_A$ allows us to express this link between the states of two transition systems [8, 9].

Definition 6 (Gluing Relation)
Let GI be a gluing invariant between SR and SA. The states $q_R \in Q_R$ and $q_A \in Q_A$ are glued, written $q_R \ \mu \ q_A$, iff $l_R(q_R) \wedge GI \Rightarrow l_A(q_A)$.

$((co{=}manual) \Leftrightarrow (co_R{=}manual)) \wedge ((co{=}stop) \Leftrightarrow ((co_R{=}stop) \wedge (seCo{=}se1)))$
$\wedge\ ((co{=}auto) \Leftrightarrow ((co_R{=}auto) \wedge (seCo{=}se1))) \wedge ((se{=}off) \Leftrightarrow (se_R{=}off))$
$\wedge\ ((se{=}not) \Leftrightarrow (se_R{=}not)) \wedge ((se{=}strong) \Leftrightarrow ((se_R{=}strong) \vee (se_R{=}minStrong)))$
$\wedge\ ((se{=}tiny) \Leftrightarrow ((se_R{=}tiny) \vee (se_R{=}maxTiny))) \wedge ((lw{=}leftDown) \Leftrightarrow (lw_R{=}leftDown))$
$\wedge\ ((lw{=}leftUp) \Leftrightarrow ((lw_R{=}leftUp) \wedge (lws{\neq}ls0))) \wedge ((rw{=}rightDown) \Leftrightarrow (rw_R{=}rightDown))$
$\wedge\ ((rw{=}rightUp) \Leftrightarrow ((rw_R{=}rightUp) \wedge (rws{\neq}rs0)))$

Figure 8: Gluing invariant GI_W

The gluing invariant GI_W linking variables of the refined system WS_R and the abstract system WS_A is given in Fig. 8. The abstract variable *co* from CO_A is linked with the concrete variables co_R and *seCo*. If *co=manual* then co_R*=manual* , if *co=stop* then co_R*=stop* and *seCo=se1*, and if *co=auto* then co_R*=auto* and *seCo=se1*. We have the same kind of links between the variables of SE_A, LW_A, RW_A and respectively, SE_R, LW_R, RW_R.

The semantic concepts of the refinement are the following.

1. In order to describe the refinement, we keep the transitions of S_R, the labels of which are in E_A (i.e. labelled by the "old" labels) and we consider the new transitions introduced during the refinement design (i.e. labelled in $E_R \setminus E_A$) as being non-observable; they are labelled by τ and called τ-transitions. Each portion of path containing τ-transitions must end by a transition labelled in E_A. Therefore, the transition refinement is either a strict refinement or a stuttering refinement (see Fig. 9).

2. In order to avoid livelocks, new transitions should not take control forever. So, the paths containing infinite sequences of τ-transitions are forbidden.

3. Moreover, new transitions should not introduce deadlocks.

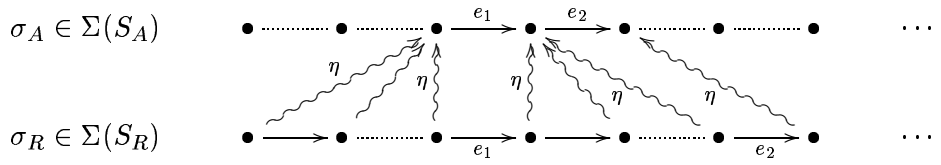

Figure 9: Path refinement

The refinement relation η is defined in [8, 9] as a restriction of μ that verifies the previous concepts.

Definition 7 (Strict Refinement Relation [8, 9])
Let SA and SR be two transition systems, and e be a label from E_A. We define the refinement relation η as the greatest binary relation included into μ and satisfying the following conditions:
1) strict transition refinement
$(q_R \ \eta \ q_A \ \wedge \ q_R \xrightarrow{e} q'_R \in T_R) \Rightarrow \exists q'_A.\ (q_A \xrightarrow{e} q'_A \in T_A \ \wedge \ q'_R \ \eta \ q'_A)$,
2) stuttering transition refinement
$(q_R \ \eta \ q_A \ \wedge \ q_R \xrightarrow{\tau} q'_R \in T_R) \Rightarrow (q'_R \ \eta \ q_A)$,
3) lack of new deadlocks
$(q_R \ \eta \ q_A \ \wedge \ q_R \nrightarrow_R) \Rightarrow (q_A \nrightarrow_A)$[1],
4) lack of τ-divergence
$q_R \ \eta \ q_A \Rightarrow \neg \ (q_R \xrightarrow{\tau} q'_R \xrightarrow{\tau} q''_R \xrightarrow{\tau} \ldots \xrightarrow{\tau} \ldots)$,
5) external non-determinism preservation
$(q_A \xrightarrow{e} q'_A \in T_A \ \wedge \ q_R \ \eta \ q_A) \Rightarrow \exists \ q'_R, q''_R, q''_A.(\ q'_R \ \eta \ q_A \ \wedge \ q'_R \xrightarrow{e} q''_R \in T_R \ \wedge \ q_A \xrightarrow{e} q''_A \in T_A \ \wedge \ q''_R \ \eta \ q''_A)$.

We say that SR *refines* SA, written $SR \sqsubseteq SA$, when the conditions above are verified between the states of SR and SA i.e. $SR \sqsubseteq SA \Leftrightarrow \forall q_R.\ (q_R \in Q_R \Rightarrow \exists q_A.\ (q_A \in Q_A \ \wedge \ q_R \ \eta \ q_A))$.

It has been shown in [8] that η is a kind of τ-simulation. It is well-known that a simulation can be computed iteratively for finite state systems. We have an algorithm based on a depth-first search enumeration of the reachability graph of the refined system. Its order is $O(|SR|)$ where $|SR| = |Q_R| + |T_R|$.

[1] We note $q \nrightarrow$ when $\forall q', e.\ (q' \in Q \wedge e \in E \Rightarrow (q \xrightarrow{e} q') \notin T)$.

3.2 Compositional Component-based Systems Refinement

In this section, the refinement semantics is fitted with synchronized component-based systems. We define a refinement relation weaker than the refinement relation presented in Section 3.1, and we clarify how τ covers the new transition labels. Then we give a compositionality theorem allowing us to compositionally ensure the refinement of the whole system from the refinement of its components.

We have shown in [21] that η is too strong to verify a component-based systems refinement. The problem is that some new deadlocks in the context-in components, could cause the refinement verification of a context-in component to fail whereas the refinement of the whole system is verified. That is why, we introduce another relation, called the weak refinement relation and written η_f. This relation is fitted to ensure the refinement verification in this case. The relation η_f uses the set $D \subseteq Q_R$ of new deadlocks which is built during the refined system exploration.

Definition 8 (Weak Refinement Relation)
*Let SA and SR be two transition systems, and e be a label from E_A. Let $D \subseteq Q_R$ (Initially $D = \varnothing$) be the set of new deadlocks. We define the weak refinement relation η_f as the greatest binary relation included into μ and satisfying the conditions **1)**, **2)**, **4)** and **5)** of Definition 7 and the following condition:*
3') old or new deadlocks
$(q_R \ \eta_f \ q_A \ \wedge \ q_R \nrightarrow_R) \Rightarrow ((q_A \nrightarrow_A) \vee ((q_A \xrightarrow{e} q'_A \in T_A) \Rightarrow (q_R \in D)))$.

We say that SR *weakly refines* SA, written $SR \sqsubseteq_D SA$, when the conditions above are verified between the states of SR and SA i.e. $SR \sqsubseteq_D SA \Leftrightarrow \forall q_R.(q_R \in Q_R \Rightarrow \exists q_A.\ (q_A \in Q_A \ \wedge \ q_R \ \eta_f \ q_A))$.

The relation η_f can be computed by an iterative algorithm with a complexity order in $O(|SR|)$ too. It is easy to see that the strict refinement relation η implies the weak refinement relation η_f when $D = \varnothing$, and vice versa. Indeed, condition ***3')*** of Definition 8 is verified for the old deadlocks, so D remains empty.

Property 1
Let SA and SR be two transition systems. Let $D \subseteq Q_R$ be the set of new deadlocks. We have $(SR \sqsubseteq SA) \Leftrightarrow (SR \sqsubseteq_D SA \ \wedge \ D = \varnothing)$.

We have to clarify how τ covers new transition labels when we deal with synchronized component-based systems. Indeed, we want to avoid the refinement verification failure when an old transition and a new one are synchronized. The problem is then to decide whether this synchronized transition has to be covered by τ or not.

Our approach is the following. On the one hand, the old transition label must be saved if it is a label of the component being considered. On the other hand, the transition must be kept out if its label is not a label of the considered component. We define a context-in τ-component to be a context-in component covered by τ. The first step of Definition 9 deletes some transitions and their labels, and the second step covers by τ the remaining transition labels.

Definition 9 (Context-in τ-component)
Let SA_1, SA_2, SR_1 and SR_2 be four components. Let SR_1^c be a context-in component. The context-in τ-component SR_1^τ is the tuple $< Q_{R1}^\tau, Q_{R01}^\tau, E_{R1}^\tau, T_{R1}^\tau, l_{R1}^\tau >$ where:
- $Q_{R1}^\tau = Q_{R1}^c$,
- $Q_{R01}^\tau = Q_{R01}^c$,
- $l_{R1}^\tau((q_1, q_2)) = l_{R1}^c((q_1, q_2))$,
- $T_{R1}^\tau = T_{R1}^c \setminus \{((q_1, q_2), (e_1, e_2), (q_1', q_2')) \mid ((q_1, q_2), (e_1, e_2), (q_1', q_2')) \in T_{R1}^c \wedge\ e_1 \in E_{R1} \setminus E_{A1}\ \wedge\ e_2 \in E_{A2}\}$,
- $E_{R1}^\tau = E_{R1}^c \setminus \{(e_1, e_2) \mid (e_1, e_2) \in E_{R1}^c \wedge e_1 \in E_{R1} \setminus E_{A1} \wedge e_2 \in E_{A2}\}$.

Then, the elements of E_{R1}^τ are covered as follows[2]*:*
- *if $(e_1, -) \in E_{R1}^\tau$ and $e_1 \in E_{R1} \setminus E_{A1}$ then $(e_1, -)\backslash\tau$,*
- *if $(e_1, e_2) \in E_{R1}^\tau$, $e_1 \in E_{R1} \setminus E_{A1}$ and $e_2 \in E_{R2} \setminus E_{A2}$ then $(e_1, e_2)\backslash\tau$,*
- *if $(e_1, e_2) \in E_{R1}^\tau$, $e_1 \in E_{A1}$ and $e_2 \in E_{R2} \setminus E_{A2}$ then $(e_1, e_2)\backslash(e_1, -)$.*

Our first result is a compositional refinement verification theorem. This theorem links the separate context-in components weak refinements with the component-based system refinement. It is based on deadlocks reduction. A state inducing a new deadlock in a component does not induce a deadlock in the whole system if there exists another component in which this state is not a deadlock state.

Definition 10 (New Deadlocks Reduction)
Let SA_1, SA_2, SR_1 and SR_2 be four components such that $SR_1^\tau \sqsubseteq_{D_1} SA_1^c$ and $SR_2^\tau \sqsubseteq_{D_2} SA_2^c$. Let Synch and Synch' be two respective synchronizations. The set $D_{1,2} \subseteq Q_{R1}^\tau \cup Q_{R2}^\tau$ of states producing new deadlocks during the weak refinement of $SR_1 \|_{Synch'} SR_2$ is defined by:

$$D_{1,2} \stackrel{def}{=} (D_1 \cap D_2) \cup (D_1 \setminus Q_{R2}^\tau) \cup (D_2 \setminus Q_{R1}^\tau)$$

We can compute the set $D_{1,\ldots,n}$ containing the new deadlocks during the weak refinement verification of $\|_{Synch'}(SR_1, \ldots, SR_n)$. It is an associative computation:

[2] We note $e_1\backslash e_2$ the relabelling of e_1 by e_2.

$D_{1,\ldots,n} = D_{(1,\ldots,n-1),n} = D_{1,(2,\ldots,n)}$. Then, *Property 1* allows us to decide the strict refinement of the whole system.

Theorem 1 (Refinement of a Synchronized Component-based System) *Let $SA_1, \ldots, SA_n$, and $SR_1, \ldots, SR_n$ be n abstract and refined components. Let Synch and Synch' be two respective synchronizations.*

$\|_{Synch'}(SR_1, \ldots, SR_n) \sqsubseteq \|_{Synch}(SA_1, \ldots, SA_n)$ *iff*
- $\forall i.\ 1 \leq i \leq n \ \Rightarrow\ SR_i^\tau \sqsubseteq_{D_i} SA_i^c$
- $D_{1,\ldots,n} = \varnothing$

A proof is given in [19].

This theorem provides a compositional refinement verification algorithm. This algorithm is based on the computation of the relation η_f for each context-in component SR_i^τ which complexity order is $O\ (|SR_i^\tau|)$. The complexity of this refinement algorithm is $O(|SR_1^\tau| + \cdots + |SR_n^\tau|)$ but it can be iteratively computed. The greatest memory space used by this algorithm computation is $max_{i=1}^{n}(|SR_i^\tau|)$, at most.

For our example, we built the refined context-in components. Figure 10 shows the refined context-in component SE_R^c. This context-in component only describes behaviours of SE_R under csw_R that are either old transitions (*seNot*, *seTiny*, *seStrong* or (*seOff,coStop*), (*seNot,coAuto*)) or new transitions (*seMaxTiny*, *seMinStrong* or (*seMaxTiny,lwSpeed1,rwSpeed1*), (*seMinStrong,lwSpeed2,rwSpeed2*)).

By Definition 5, we have $WS_R = CO_R^c \uplus SE_R^c \uplus LW_R^c \uplus RW_R^c$ over the set of variables $Var_R = \{co_R, seCo, lw_R, lws, rw_R, rws, se_R\}$. The weak refinement is verified for each context-in component. We have $SE_R^c \sqsubseteq_{D_{SE}} SE_A^c$ where the set D_{SE} contains two new deadlocks q_6^c and q_8^c (see Fig. 10). By Definition 10, we reduce D_{SE} and we show that $D_{CO,SE,LW,RW}$ is empty. By Theorem 1, we conclude that the wipers system WS_R refines WS_A.

The refinement relation provides a formal framework for verifying system properties. When the refinement is verified, most properties that have been verified on the abstract system are preserved on the refined one.

4 Compositional Property Verification

In this section, we propose a method to verify a class of safety properties on synchronized component-based systems. These properties can be expressed as the unreachability of a set of states. If this set is not reachable in the abstract system, then the

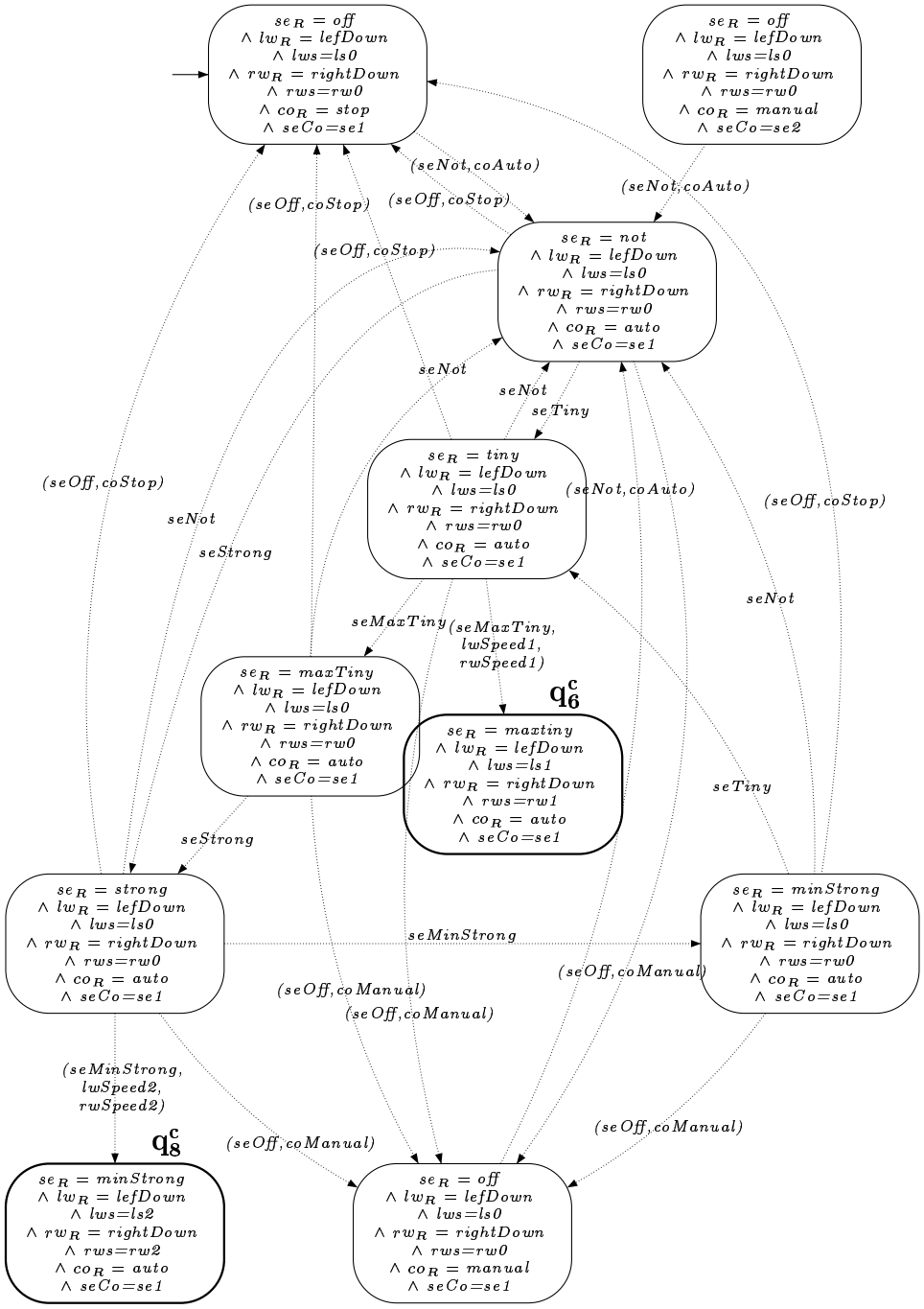

Figure 10: SE_R^c

refinement relation ensures the unreachability of the corresponding set in the refined system.

Since we introduce new details in the refinement, the abstract system is usually small in number of states while the refined system is likely to be very large. We want to exploit the refinement approach enriched by a component paradigm. It allows us to avoid the model-checking blow-up by verifying safety properties in a compositional way.

The method we propose here is based on Definition 5. The idea is to verify the reachability among the context-in components S_i^c instead of exploring the whole system $\|_{Synch}(S_1, \ldots, S_n)$. Actually, the state space and the transition relation of a context-in component are less important than the entire system ones. We reduce the number of possible behaviours during the system exploration, and, consequently, we postpone the state space explosion problem.

Let $\|_{Synch}(S_1, \ldots, S_n)$ be a synchronized parallel composition under $Synch$. The reachability problem for $\|_{Synch}(S_1, \ldots, S_n)$ is the following decision problem.
Input: n context-in components $S_1^c, \ldots, S_n^c$ and a target state q_t^c s.t. $q_t^c \in \bigcup_{i=1}^n Q_i^c$.
Reachability problem (RP): Determine whether q_t^c is in $\{q^c \mid q^c \in \bigcup_{i=1}^n Q_i^c \ \wedge \ \exists q_0^c.\ (q_0^c \in \bigcup_{i=1}^n Q_{0i}^c \ \wedge \ \exists w.\ (w \in (\bigcup_{i=1}^n E_i^c)^* \ \wedge \ (q_0^c, w, q^c) \in (\bigcup_{i=1}^n T_i^c)^*))\}$, where $(\bigcup_{i=1}^n T_i^c)^*$ is the reflexive and transitive closure of the transition relation $\bigcup_{i=1}^n T_i^c$.

Theorem 2 (RP for a Synchronized Component-based System)
There exists an algorithm to decide the reachability problem for a synchronized component-based system $\|_{Synch}(S_1, \ldots, S_n)$ *whose complexity order is in* $O(|S_1^c| + \cdots + |S_n^c|)$.

In practice, to verify the reachability of a target state q_t^c in a context-in component, we apply the backward-reachability analysis from q_t^c. During this exploration, the set Q_D of the deadlock states is built. If no initial state is reachable, we choose another context-in component, and, often, a new target state among the deadlock states (in Q_D), in order to continue the reachability analysis from this state. We can choose a new target, since the state q_t^c is reachable from this new state. We stop when either an initial state $q_0^c \in \bigcup_{i=1}^n Q_{0i}^c$ is reached or exploration of all possible choices is already done.

The verification algorithm below formally presents the compositional reachability analysis.

Algorithm 1 (Compositional Reachability)

```
1  Input
2     (Q_0^c = ⋃_{i=1}^n Q_{0i}^c        (*Initial state space*)
3     q_i^c ∈ ⋃_{i=1}^n Q_{0i}^c         (*Target state*)
4     CI ⊆ {S_1^c, ..., S_n^c}           (*A set of context-in compo-ts*)
5  Result
6     reach: boolean     (*true if q_i^c is reachable*)
7  Variables
8     Q_D^c, Q_ND^c, Q_suc^c, Q_pre^c ⊆ ⋃_{i=1}^n Q_{0i}^c
9     S_i^c ∈ CI
10    (q_i^c ∈ ⋃_{i=1}^n Q_{0i}^c
11    tested ⊆ ⋃_{i=1}^n Q_{0i}^c × {S_1^c, ..., S_n^c}
12    end, possible, deadlock: booleen
13 Begin
14    Q_D^c := {q_i^c}
15    end := false
16    reach := false
17    WHILE (end = false) DO
18       Choice (Q_D^c, CI, tested, possible, q_i^c, S_i^c)
19       IF (possible = false) DO
20          end := true
21       ELSE
22          tested := tested ∪{(q_i^c, S_i^c)}
23          deadlock := false
24          Q_suc^c := {q_i^c}
25          Q_pre^c := {}
26          WHILE (deadlock = false) DO
27             Predecessors (Q_suc^c, S_i^c, Q_pre^c, Q_ND^c)
28             Q_D^c := Q_D^c ∪ Q_ND^c
29             IF (Q_pre^c = {}) DO
30                deadlock := true
31             ELSE
32                IF (Q_0^c ∩ Q_pre^c ≠ {}) DO
33                   deadlock := true
34                   end := true
35                   reach := true
36                ELSE
37                   Q_suc^c := Q_pre^c
38                FI
39             FI
40          ENDWHILE
41       FI
42    ENDWHILE
43 End
```

This algorithm uses the following procedures.
- Predecessors(Input: Q^c_{suc}, S^c_i, Output: Q^c_{pre}, Q^c_{ND}),
- Choice(Input: Q^c_D, CI, *tested*, Output: *possible*, q^c_i, S^c_i).

The first procedure computes Q^c_{pre} in S^c_i, i.e., the set of prececessors of Q^c_{suc}. The states in Q_{suc} without predecessors are put in a set Q^c_{ND} of new deadlocks. Formally we have $Q^c_{pre} \stackrel{\text{def}}{=} \{q^c \mid (q^c, \alpha, q^{c\prime}) \in T^c_i \wedge q^{c\prime} \in Q^c_{suc}\}$. The second procedure chooses, among both states in Q^c_D and context-in components in CI, the most convenient and not tested yet pair (q^c_i, S^c_i). To be efficient, this choice has to be based on some heuristics. For example, the strong dependency analysis of [22] between components can be used for an efficient reachability analysis of their synchronized composition.

This compositional reachability algorithm becomes very interesting when combined with the refinement verification approach. Indeed, all the components in the context are computed during the refinement verification.

Theorem 3 (Correctness of Algorithm 1)
Algorithm 1 eventually terminates and indicates whether the target state is reachable ($reach = true$) or not ($reach = false$).

Proof idea. The computation in the loop $WHILE$ ($deadlock = false$) (lines 26-42) stops. The predecessors computation stops either when an initial state is reached (line 33), or when there are no predecessors (line 30). The computation of the loop $WHILE$ ($end = false$) (lines 17-42) eventually terminates. The variable *end* becomes *true* since either an initial state is reached (line 34), or there is no other choice for a pair of a context-in component and a deadlock state (line 20). The only

possibility to have the variable *reach* equal to *true* (line 35) is to reach an initial state.

For our running wind-screen wipers example, we want some states to be forbidden. For instance, states where the wipers system can move only one of the wind-screen wipers, must be unreachable. Then we exploit the refinement to ensure this property for the refined system.

Consider q_t^c, a state such that $l(q_t^c) \stackrel{\text{def}}{=} (lw = leftDown \ \wedge \ rw = rightUp \ \wedge \ co = manual \ \wedge \ se = off)$. We want to ensure the unreachability of the state q_E^c, written $\Box(\neg \ q_t^c)$ in $PLTL$. By Algorithm 1, it is possible to show that q_t^c is not reachable in the abstract system WS_A. As the system WS_R refines the system WS_A (as explained in Section 3), the refined system states corresponding to q_t^c by the gluing invariant GI_W are unreachable too.

5 *SynCo*: a Tool Verifying the Synchronized Component-based Systems Refinement

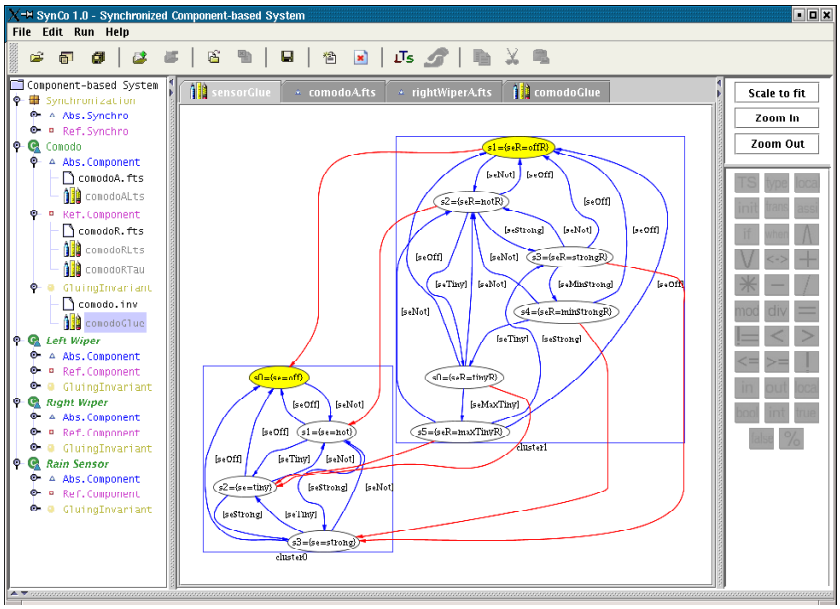

Figure 11: The tool *SynCo*

Theorem 1 in Section 3 gives the required conditions to ensure the strict refinement of a whole system from the weak refinements of its synchronized components, and vice versa. This way, we have an algorithm to verify the strict refinement of a synchronized component-based system. The most important advantage of this refinement verification is that it is not necessary to build the whole system to verify the refinement.

```
Transition System
type CONTROLR = {stopR, autoR, manualR}
type SELECTR = {se1, se2}
local coR : CONTROLR
local seCo :  SELECT
Initially (coR = stopR) ∧ (seCo = se1)
Transition coStop :
enable (coR != stopR) ;
assign coR := stopR, seCo := se1
Transition coAuto :
enable (coR != autoR) ;
assign coR := autoR, seCo := se1
Transition coManual :
enable (coR = stopR) ∨ (coR = autoR) ;
assign coR := manualR, seCo := se1
Transition coSelect :
enable (coR = manualR) ∧ (seCo = se1) ;
assign seCo := se2
```

Figure 12: CO_R in STeP

We are developing *SynCo* (for *Synchronized Component-based System Analyser*)[3]; a tool implementing with Java the synchronized component refinement verification (Figure 11 show a capture of its interface). We specify the components as transition systems using the STeP's Fair Transition Systems syntax [23, 10]. Each transition is specified with its activation condition (`enable`) and its assignments (`assign`). Figure 12 gives the STeP specification of the component CO_R.

The context-in components are built automatically from both STeP and synchronization specifications. Each component or each context-in component can be visualised as an automaton using the Java package *Grappa* from the toolkit *Graphviz*[4].

Theorem 1 is implemented in *SynCo* to verify the refinement of a synchronized component-based system. We compute the relation η_f for each context-in τ-component using a kind of depth-first search enumeration of the reachability graph of this context-in τ-component. Then, we conclude about the refinement of the whole system. Figure 11 shows the relation η_f between a refined component and an abstract component.

The refinement verification of a simple system can be done with *SynCo* too. We assume that the simple system is the one component of a synchronized component-based system. Obviously, the new deadlocks set is empty by Property 1.

With this tool the refinement verification of the wind-screen wipers system and of an industrial robot has been done. Thanks to *SynCo* we can measure the efficiency of our method. Tables 13 and 14 give the results of our tests. For the wind-screen wipers system WS_R both the states and the transitions are decreased by a third between the whole system and the greatest context-in component (see Fig. 13). We

[3] *SynCo* home page: `http://lifc.univ-fcomte.fr/~lanoix/synco.html`
[4] Graphviz official page: `http://www.research.att.com/sw/tools/graphviz`

	CO_R^c	SE_R^c	LW_R^c	RW_R^c	WS_R
$\|Q_R\|$	8	10	18	18	20
$\|T_R\|$	17	25	20	20	47
$\|SR\|$	25	35	38	38	67

Figure 13: The wipers system

	CL_R^c	LI_R^c	HA_R^c	RO_R
$\|Q_R\|$	19	13	19	25
$\|T_R\|$	21	13	20	40
$\|SR\|$	40	26	39	65

Figure 14: The robot

have similar results in Fig. 14 for the industrial robot RO_R moving pieces thanks to a clip CL_R, a handle HA_R and a lift LI_R. We plan to test *SynCo* on more complex examples.

6 Conclusion and Perspectives

Compositional design and refinement paradigms are very convenient to specify and to verify large embedded reactive systems such as automatic trains, car driving assistance systems or communication protocols. In this paper, we propose a technique to take advantages of both paradigms. This technique is based on a specific notion of refinement relation defined in the style of Milner-Park simulation relation between concrete and abstract finite transition systems.

The contributions of this paper are the following. On the one hand, we want the refinement paradigm to be compatible with the compositional specification. For that, we propose a new compositional specification method that allows us to ensure the refinement of the whole system from the refinement of its context-in components, and vice versa. Although context-in components seem comparable with the whole system, they are generally smaller than the whole system. On the other hand, we want to avoid the model-checking blow-up by verifying a class of safety properties in a compositional way. For that, we give an algorithm to ensure the unreachability of a set of states of a synchronized component-based system. Moreover, this algorithm is very interesting when combined with the compositional refinement verification algorithm.

A tool that implements the compositional refinement verification has been implemented in Java. This tool, *SynCo*, allows us to measure the efficiency of our approach, and to test it to real industrial examples. We are currently working on extending this tool to incorporate the compositional reachability algorithm. In addition, we are going to use the strong dependency analysis of [22] as heuristics to efficiently choose a pair (q_i^c, S_i^c). As far as we know, other compositional tools also implement a reachability compositional verification (see for instance [15, 22]). How-

ever, they were designed for a very different purpose than *SynCo* and, thus, do not implement compositional refinement verification.

One of the consequences of the composition theorem in [25] is the limitation of the compositional verification. It has been shown that the composition theorem fails for very simple parallel operators if there is a property expressing that "there is a path such that all the nodes of the path have a property p". Moreover, we have established in [14] that the strict refinement relation preserves the abstract system $PLTL$ properties, i.e. safety and liveness properties because of Conditions ***3)*** and ***4)*** of Definition 7. This preservation partially avoids the combinatorial explosion problem. Indeed, a property proof on an abstract system can be reused as a hypothesis for its verification on the refined system. That is why we are interested in a compositional verification of $PLTL$ properties.

More generally, we are going to use the works on property preservation [14], modular verification [24, 17], and dynamic property refinement [6, 7] in the framework of the component-based system specification and refinement presented in this paper. We hope this will allow us to verify a large spectrum of properties in a compositional way.

Acknowledgement

We are very grateful to Steve Campos and Caroline Lasalle for their participation in the development of *SynCo*.

References

[1] J.-R. Abrial. *The B Book.* Cambridge University Press - ISBN 0521-496195, 1996.

[2] J.-R. Abrial. Constructions d'automatismes industriels avec B. In *Congrès AFADL*, ONERA-CERT - Toulouse, France, May 1997. Invited lecture.

[3] J.-R. Abrial. Discrete system models. Version 1.1, February 2002.

[4] J.-R. Abrial and L. Mussat. Introducing dynamic constraints in B. In *Second Conference on the B method, France*, volume 1393 of *LNCS*, pages 83–128. Springer Verlag, April 1998.

[5] P. Behm, P. Desforges, and J.M. Meynadier. MÉTÉOR: An industrial success in formal development. In *Second conference on the B method*, volume 1393 of

Lecture Notes in Computer Science, Montpellier, France, April 1998. Springer Verlag. Invited lecture.

[6] F. Bellegarde, C. Darlot, J. Julliand, and O. Kouchnarenko. Reformulate dynamic properties during B refinement and forget variants and loop invariants. In *Proc. First Int. Conf. ZB'2000, York, Great Britain*, volume 1878 of *LNCS*, pages 230–249. Springer-Verlag, September 2000.

[7] F. Bellegarde, C. Darlot, J. Julliand, and O. Kouchnarenko. Reformulation: a way to combine dynamic properties and B refinement. In *In Proc. Int. Conf. Formal Method Europe'01, Berlin, Germany*, volume 2021 of *LNCS*, pages 2–19. Springer Verlag, March 2001.

[8] F. Bellegarde, J. Julliand, and O. Kouchnarenko. Ready-simulation is not ready to express a modular refinement relation. In *Proc. Int. Conf. on Fundamental Aspects of Software Engineering, FASE'2000*, volume 1783 of *LNCS*, pages 266–283. Springer-Verlag, April 2000.

[9] F. Bellegarde, J. Julliand, and O. Kouchnarenko. Synchronized parallel composition of event systems in *B*. In D. Bert, J. P. Bowen, M. C. Henson, and K. Robinson, editors, *ZB 2002 : Formal specification and development in Z and B*, volume 2272 of LNCS, pages 436–457. Springer-Verlag, 2002.

[10] N. Bjørner, A. Browne, M. Colón, B. Finkbeiner, Z. Manna, M. Pichora, H. B. Sipma, and T. E. Uribe. *STeP - The Stanford Temporal Prover - Educational Release - User's Manual*. Computer Science Department - Stanford University, Stanford, California 94305, july 1998.

[11] P. Bontron and M.-L. Potet. Automatic construction of validated B components from structured developments. In *Proc. First Int. Conf. ZB'2000, York, Great Britain*, volume 1878 of *LNCS*, pages 127–147. Springer-Verlag, September 2000.

[12] M. J. Butler. csp2B: A practical approach to combining CSP and B. *Formal Aspects of Computing*, 12:182–198, 2000.

[13] M. J. Butler and M. Waldén. *Program Development by Refinement (Case Studies Using the B Method)*, chapter Parallel Programming with the B Method. Springer, 1999.

[14] C. Darlot, J. Julliand, and O. Kouchnarenko. Refinement preserves PLTL properties. In D. Bert, J. P. Bowen, S. C. King, and M. Walden, editors, *ZB'2003:*

Formal Specification and Development in Z and B, volume 2651 of *LNCS*, Turku, Finland, June 2003. Springer-Verlag.

[15] Hubert Garavel, Frédéric Lang, and Radu Mateescu. An overview of CADP 2001. Technical Report RT-254, INRIA, December 2001.

[16] J. Julliand, F. Bellegarde, and B. Parreaux. De l'expression des besoins à l'expression formelle des propriétés dynamiques. *Technique et Science Informatiques*, 18(7), 1999.

[17] J. Julliand, P-A. Masson, and H. Mountassir. Vérification par model-checking modulaire des propriétés dynamiques introduites en B. *Technique et Science Informatiques*, 20(7):927–957, 2001.

[18] J. Julliand, P.A. Masson, and H. Mountassir. Modular verification of dynamic properties for reactive systems. In *International Workshop on Integrated Formal Methods (IFM'99)*, York, Great Britain, 1999.

[19] O. Kouchnarenko and A. Lanoix. Refinement and verification of synchronized component-based systems. INRIA Research Report, June 2003. To appear.

[20] L. Lamport. Specifying concurrent systems with TLA+. In *Calculational System Design*, Amsterdam, 1999. IOS Press.

[21] A. Lanoix. Raffinement de systèmes à composants synchronisés et leur vérification. Mémoire de DEA - UFR Sciences et Techniques - Université de Franche-Comté, Septembre 2002.

[22] J. Lind-Nielsen, H. R. Andersen, H. Hulgaard, G. Behrmann, K. Kristoffersen, and K. G. Larsen. Verification of large state/event systems using compositionality and dependency analysis. *Formal Methods in System Design*, 18(1):5–23, January 2001.

[23] Z. Manna, N. Bjørner, A. Browne, E. Chang, M. Colón, Luca de Alfaro, H. Devarajan, A. Kapur, J. Lee, H. B. Sipma, and T. E. Uribe. STeP - the stanford temporal prover. *Theory and Practice of Software Developpement*, 915 of LNCS:793–794, May 1995.

[24] P.-A. Masson, H. Mountassir, and J. Julliand. Modular verification for a class of PLTL properties. In T. Santen W. Grieskamp and B. Stoddart, editors, *2nd international conference on Integrated Formal Methods, IFM 2000*, volume 1945 of *LNCS*, pages 398–419. Springer-Verlag, November 2000.

[25] A. Rabinovich. On compositional method and its limitations. Technical report, University of Edinburgh, Research Report EDI-INF-RR-0035, 2001.

[26] S. Schneider and H. Treharne. Communicating B machines. In D. Bert, J. P. Bowen, M. C. Henson, and K. Robinson, editors, *ZB 2002 : Formal specification and development in Z and B*, volume 2272 of LNCS, pages 416–435. Springer-Verlag, 2002.

[27] E. Sekerinski. *Program Development by Refinement (Case Studies Using the B Method)*, chapter Production Cell. Springer, 1999.

[28] H. Treharne, J. Draper, and S. Schneider. Test case preparation using a prototype. In *Second conference on the B method*, LNCS 1393, pages 293–312, Montpellier, France, April 1998. Springer Verlag.

[29] H. Treharne and S. Schneider. Using a process algebra to control B OPERATIONS. In *IFM'99 1st International Conference on Integrated Formal Methods*, pages 437–457, York, 1999. Springer-Verlag.

A Proof of Theorem 1

We prove that $SR_1 \| _{Synch_R} SR_2 \sqsubseteq SA_1 \| _{Synch_A} SA_2 \quad [R]$ iff $\begin{cases} SR_1^\tau \sqsubseteq_{D_1} SA_1^c, & [a] \\ SR_2^\tau \sqsubseteq_{D_2} SA_2^c, & [b] \\ D_{12} = \varnothing. & [c] \end{cases}$

It can be extended to n components.
For that, we show that items $[a]$, $[b]$ and $[c]$ imply $[R]$ (see Section A.1). Then, we show that $[R]$) implies $[a]$, $[b]$ and $[c]$ (see Section A.2).

Let Q_R^c be the state set of $SR_1 \| _{Synch_R} SR_2$, and Q_A^c the state set of $SA_1 \| _{Synch_A} SA_2$. By Definition 2, we have $Q_R^c = Q_{R1}^c \cup Q_{R2}^c$ and $Q_A^c = Q_{A1}^c \cup Q_{A2}^c$.

A.1 $[a] \wedge [b] \wedge [c] \Rightarrow [R]$

Let $M \stackrel{\text{def}}{=} \{(q_R^c, q_A^c) \mid ((q_R^c \in Q_{R1}^c \wedge q_A^c \in Q_{A1}^c) \vee (q_R^c \in Q_{R2}^c \wedge q_A^c \in Q_{A2}^c)) \wedge q_R^c \, \eta_f \, q_A^c\}$.

We show that M verifies the conditions of the refinement relation η (Definition 7) using Lemmas 1, 2, 3, 4 and 5.

By $[c]$ we have $D_{12} = \varnothing$. As $D_{12} = (D_1 \cap D_2) \cup (D_1 \setminus Q_{R2}^c) \cup (D_2 \setminus Q_{R1}^c)$ (Definition 10), we have $[c'] : (D_1 \cap D_2 = \varnothing) \wedge (D_1 \setminus Q_{R2}^c = \varnothing) \wedge (D_2 \setminus Q_{R1}^c = \varnothing)$.

Lemma 1 (Strict Transition Refinement Condition)
Assume that $q_R^c \stackrel{\alpha}{\rightarrow} q_R^{c\prime}$ and $\alpha \in E_{R1}^c \cup E_{R2}^c$; we must prove that there exists $q_A^{c\prime}$ such that $q_A^c \stackrel{\alpha}{\rightarrow} q_A^{c\prime}$ and $q_R^{c\prime} \eta \; q_A^{c\prime}$.

Proof on the form of α. There are three cases.

- *$\alpha = (e_1, -)$. By Definition 4, we have $(q_R^c, (e_1, -), q_R^{c\prime}) \in T_{R1}^c$, $q_R^c \in Q_{R1}^c$ and $q_R^{c\prime} \in Q_{R1}^c$. Since $SR_1^\tau \sqsubseteq SA_1^c$, we obtain that there exists q_A^c such that $q_A^c \in Q_{A1}^c$ and $q_R^c \eta \; q_A^c$. By condition 1 for $SR_1^\tau \sqsubseteq SA_1^c$, there exists $q_A^{c\prime}$ such that $q_A^{c\prime} \in Q_{A1}^c$, $q_A^c \stackrel{(e_1,-)}{\rightarrow} q_A^{c\prime}$ and $q_R^{c\prime} \eta \; q_A^{c\prime}$. The states q_A^c and $q_A^{c\prime}$ belong to Q_A^c because of $Q_A^c = Q_{A1}^c \cup Q_{A2}^c$.*

- *$\alpha = (-, e_2)$. By Definition 4, we have $(q_R^c, (-, e_2), q_R^{c\prime}) \in T_{R2}^c$, $q_R^c \in Q_{R2}^c$ and $q_R^{c\prime} \in Q_{R2}^c$. Since $SR_2^\tau \sqsubseteq SA_2^c$, we obtain that there exists q_A^c such that $q_A^c \in Q_{A2}^c$ and $q_R^c \eta \; q_A^c$. By condition 1 for $SR_2^\tau \sqsubseteq SA_2^c$, there exists $q_A^{c\prime}$ such that $q_A^{c\prime} \in Q_{A2}^c$, $q_A^c \stackrel{(-,e_2)}{\rightarrow} q_A^{c\prime}$ and $q_R^{c\prime} \eta \; q_A^{c\prime}$. The states q_A^c and $q_A^{c\prime}$ belong to Q_A^c because of $Q_A^c = Q_{A1}^c \cup Q_{A2}^c$.*

- *$\alpha = (e_1, e_2)$. By Definition 4, we have $(q_R^c, (e_1, e_2), q_R^{c\prime}) \in T_{R1}^c$, $q_R^c \in Q_{R1}^c$ and $q_R^{c\prime} \in Q_{R1}^c$. Since $SR_1^\tau \sqsubseteq SA_1^c$, we obtain that there exists q_A^c such that $q_A^c \in Q_{A1}^c$*

and $q_R^c \eta\ q_A^c$. *By condition 1 for* $SR_1^\tau \sqsubseteq SA_1^c$, *there exists* $q_A^c{}'$ *such that* $q_A^c{}' \in Q_{A1}^c$, $q_A^c \stackrel{(e_1,-)}{\rightarrow} q_A^c{}'$ *and* $q_R^c{}' \eta\ q_A^c{}'$. *The states* q_A^c *and* $q_A^c{}'$ *belong to* Q_A^c *because of* $Q_A^c = Q_{A1}^c \cup Q_{A2}^c$.

Lemma 2 (Stuttering Transition Refinement Condition)
Assume that $q_R^c \stackrel{\tau}{\rightarrow} q_R^c{}'$; *we must show that* $q_R^c{}' \eta\ q_A^c$.

Proof on the form of labels covered by τ. *There are three cases.*

- $(e_1', -)\backslash\tau$. *By Definition 4, we have* $(q_R^c, (e_1', -)\backslash\tau, q_R^c{}') \in T_{R1}^c$, $q_R^c \in Q_{R1}^c$ *and* $q_R^c{}' \in Q_{R1}^c$. *Since* $SR_1^\tau \sqsubseteq SA_1^c$, *there exists* q_A^c *such that* $q_A^c \in Q_{A1}^c$ *and* $q_R^c \eta\ q_A^c$. *By condition 2 for* $SR_1^\tau \sqsubseteq SA_1^c$, *we obtain that* $q_R^c{}' \eta\ q_A^c$.

- $(-, e_2')\backslash\tau$. *by Definition 4, we have* $(q_R^c, (-, e_2')\backslash\tau, q_R^c{}') \in T_{R2}^c$, $q_R^c \in Q_{R2}^c$ *and* $q_R^c{}' \in Q_{R2}^c$. *Since* $SR_2^\tau \sqsubseteq SA_2^c$, *there exists* q_A^c *such that* $q_A^c \in Q_{A2}^c$ *and* $q_R^c \eta\ q_A^c$. *By condition 2 for* $SR_2^\tau \sqsubseteq SA_2^c$, *we obtain that* $q_R^c{}' \eta\ q_A^c$.

- $(e_1', e_2')\backslash\tau$. *By Definition 4, we have* $(q_R^c, (e_1', e_2')\backslash\tau, q_R^c{}') \in T_{R1}^c$, $q_R^c \in Q_{R1}^c$ *and* $q_R^c{}' \in Q_{R1}^c$. *Since* $SR_1^\tau \sqsubseteq SA_1^c$, *there exists* q_A^c *such that* $q_A^c \in Q_{A1}^c$ *and* $q_R^c \eta\ q_A^c$. *By condition 2 for* $SR_1^\tau \sqsubseteq SA_1^c$, *we obtain that* $q_R^c{}' \eta\ q_A^c$.

Lemma 3 (Lack of New Deadlocks Condition)
Assume that $q_R^c \nrightarrow$; *we must prove that* $q_A^c \nrightarrow$.

Proof on the structure of the state set. There are three cases.

- $q_R^c \in Q_{R1}^c \setminus Q_{R2}^c$. *As we have* $q_R^c \eta_f\ q_A^c$, *condition 3 in* $SR_1^\tau \sqsubseteq_{D_1} SA_1^c$ *gives:*
  - *either* $q_A^c \nrightarrow$ *and we are done,*
  - *or* $q_A^c \stackrel{e}{\rightarrow}_A q_A^c{}'$ *and* $q_R^c \in D_1$. *As* $q_R^c \notin Q_{R2}^c$, *we have a contradiction with* $D_1 \setminus Q_{R2}^c = \varnothing$.

- $q_R^c \in Q_{R2}^c \setminus Q_{R1}^c$. *As we have* $q_R^c \eta_f\ q_A^c$, *condition 3 in* $SR_2^\tau \sqsubseteq_{D_2} SA_2^c$ *gives:*
  - *either* $q_A^c \nrightarrow$ *and we are done,*
  - *or* $q_A^c \stackrel{e}{\rightarrow}_A q_A^c{}'$ *and* $q_R^c \in D_2$. *As* $q_R^c \notin Q_{R1}^c$, *we have a contradiction with* $D_2 \setminus Q_{R1}^c = \varnothing$.

- $q_R^c \in Q_{R1}^c \cap Q_{R2}^c$. *As we have* $q_R^c \eta_f\ q_A^c$, *condition 3 in* $SR_1^\tau \sqsubseteq_{D_1} SA_1^c$ *gives:*
  - *either* $q_A^c \nrightarrow$ *and we are done,*
  - *or* $q_A^c \stackrel{e}{\rightarrow}_A q_A^c{}'$ *and* $q_R^c \in D_1$.

Moreover, condition 3 in $SR_2^\tau \sqsubseteq_{D_2} SA_2^c$ *gives:*
  - *either* $q_A^c \nrightarrow$ *and we are done,*
  - *or* $q_A^c \stackrel{e}{\rightarrow}_A q_A^c{}'$, $q_R^c \in D_1$ *and* $q_R^c \in D_2$. *We have a contradiction with* $D_1 \cap D_2 = \varnothing$.

Lemma 4 (Lack of τ-divergence Condition)
We must prove that there is no $(q_R^c \xrightarrow{\tau} q_R^c{}' \xrightarrow{\tau} q_R^c{}'' \xrightarrow{\tau} \ldots \xrightarrow{\tau} \ldots)$ *in* M.

Proof by contradiction. We suppose that $(q_R \xrightarrow{\tau} q_R^c{}' \xrightarrow{\tau} q_R^c{}'' \xrightarrow{\tau} \ldots \xrightarrow{\tau} \ldots)$ *exists.*

- $q_R^c, q_R^c{}', q_R^c{}'', \ldots \in Q_{R1}^c$. *We have a contradiction with condition 4 for* $SR_1^c \sqsubseteq SA_1^c$.
- $q_R^c, q_R^c{}', q_R^c{}'', \ldots \in Q_{R2}^c$. *We have a contradiction with condition 4 for* $SR_2^c \sqsubseteq SA_2^c$.
- $\exists i.(q_R^{c\ (i)} \in Q_{R1}^c \wedge q_R^{c\ (i+1)} \in Q_{R2}^c)$. *Inconsistence with condition 3 for* $SR_1^\tau \sqsubseteq SA_1^c$.

Lemma 5 (Non-determinism Preservation Condition)
Assume that $q_A^c \xrightarrow{\alpha} q_A^c{}'$; *we must show that there exists* $q_R^c{}'$, $q_R^c{}''$ *and* $q_A^c{}''$ *such that* $q_R^c{}' \eta\ q_A^c$, $q_R^c{}' \xrightarrow{\alpha} q_R^c{}''$, $q_A^c \xrightarrow{\alpha} q_A^c{}''$, *and* $q_R^c{}'' \eta\ q_A^c{}''$.

Proof on the form of α. *There are three cases.*

- $\alpha = (e_1, -)$. *By Definition 4, we have* $(q_A^c, (e_1, -), q_A^c{}') \in T_{A1}^c$, $q_A^c \in Q_{A1}^c$ *and* $q_A^c{}' \in Q_{A1}^c$. *Since* $SR_1^\tau \sqsubseteq SA_1^c$, *there exists* q_R^c *such that* $q_R^c \in Q_{R1}^c$ *and* $q_R^c \eta\ q_A^c$. *By condition 5 for* $SR_1^\tau \sqsubseteq SA_1^c$, *we obtain that there exist* $q_R^c{}'$, $q_R^c{}''$ *and* $q_A^c{}''$ *such that* $q_R^c{}' \eta\ q_A^c$, $q_R^c{}' \xrightarrow{\alpha} q_R^c{}''$, $q_A^c \xrightarrow{\alpha} q_A^c{}''$ *and* $q_R^c{}'' \eta\ q_A^c{}''$.

- $\alpha = (-, e_2)$. *By Definition 4, we have* $(q_A^c, (-, e_2), q_A^c{}') \in T_{A2}^c$, $q_A^c \in Q_{A2}^c$ *and* $q_A^c{}' \in Q_{A2}^c$. *Since* $SR_2^\tau \sqsubseteq SA_2^c$, *there exists* q_R^c *such that* $q_R^c \in Q_{R2}^c$ *and* $q_R^c \eta\ q_A^c$. *By condition 5 for* $SR_2^\tau \sqsubseteq SA_2^c$, *we obtain that there exist* $q_R^c{}'$, $q_R^c{}''$ *and* $q_A^c{}''$ *such that* $q_R^c{}' \eta\ q_A^c$, $q_R^c{}' \xrightarrow{\alpha} q_R^c{}''$, $q_A^c \xrightarrow{\alpha} q_A^c{}''$ *and* $q_R^c{}'' \eta\ q_A^c{}''$.

- $\alpha = (e_1, e_2)$. *By Definition 4, we have* $(q_A^c, (e_1, e_2), q_A^c{}') \in T_{A1}^c$, $q_A^c \in Q_{A1}^c$ *and* $q_A^c{}' \in Q_{A2}^c$. *Since* $SR_1^\tau \sqsubseteq SA_1^c$, *there exists* q_R^c *such that* $q_R^c \in Q_{R1}^c$ *and* $q_R^c \eta\ q_A^c$. *By condition 5 for* $SR_1^\tau \sqsubseteq SA_1^c$, *we obtain that there exist* $q_R^c{}'$, $q_R^c{}''$ *and* $q_A^c{}''$ *such that* $q_R^c{}' \eta\ q_A^c$, $q_R^c{}' \xrightarrow{\alpha} q_R^c{}''$, $q_A^c \xrightarrow{\alpha} q_A^c{}''$ *and* $q_R^c{}'' \eta\ q_A^c{}''$.

A.2 $[R] \Rightarrow [a] \wedge [b] \wedge [c]$

Property 1 gives immediately $D_{12} = \varnothing$ $[c]$ and $SR_1 \|_{Synch'} SR_2 \sqsubseteq_{D_{12}} SA_1 \|_{Synch} SA_2$ $[R']$. We show that $[R']$ implies $[a]$ and $[b]$.

Let $M_1 \stackrel{\text{def}}{=} \{(q^c_{R1}, q^c_{A1}) \mid (q^c_{R1} \in Q^c_{R1} \wedge q^c_{A1} \in Q^c_{A1}\}$
and $M_2 \stackrel{\text{def}}{=} \{(q^c_{R2}, q^c_{A2}) \mid (q^c_{R2} \in Q^c_{R2} \wedge q^c_{A2} \in Q^c_{A2}\}$ be two sets
with $\forall q^c_R.(q^c_R \in Q^c_{R1} \cup Q^c_{R2} \Rightarrow \exists q^c_A.(q^c_A \in Q^c_{A1} \cup Q^c_{A2} \wedge q^c_R \ \eta_f \ q^c_A))$.

We show that M_1 verifies the conditions of Definition 8 using Lemmas 6, 7, 8, 9 and 10. The proof is the same for M_2.

As $D_{12} = \varnothing$, we always have $[c'] : (D_1 \cap D_2 = \varnothing) \wedge (D_1 \setminus Q^c_{R2} = \varnothing) \wedge (D_2 \setminus Q^c_{R1} = \varnothing)$.

Lemma 6 (Strict Transition Refinement Condition)
Assume that $q^c_{R1} \stackrel{\alpha}{\rightarrow} q^c_{R1}{}'$; we must prove that there exists $q^c_{A1}{}'$ such that $q^c_{A1} \stackrel{\alpha}{\rightarrow} q^c_{A1}{}'$ and $q^c_R{}' \eta \ q^c_A{}'$.

Proof on the form of α. There are three cases.

- $\alpha = (e_1, -)$ or $\alpha = (e_1, e_2)$. By Definition 4, we have $q^c_{R1} \in Q^c_{R1}$ and $q^c_{R1}{}' \in Q^c_{R1}$. Since $q^c_{R1} \ \eta_f \ q^c_{A1}$ for the whole system, we obtain that there exist q^c_{A1} and $q^c_{A1}{}'$ such that $q^c_{A1}, q^c_{A1}{}' \in Q^c_A$, $q^c_{A1} \stackrel{\alpha}{\rightarrow} q^c_{A1}{}'$ and $q^c_R{}' \eta \ q^c_A{}'$. By Definition 4, we have q^c_{A1} and $q^c_{A1}{}'$ belonging to Q^c_{A1}. We conclude that $(q^c_{R1}, q^c_{A1}) \in M_1$.

- $\alpha = (-, e_2)$. Impossible because of Definition 4.

Lemma 7 (Stuttering Transition Refinement Condition)
Assume $q^c_{R1} \stackrel{\tau}{\rightarrow} q^c_{R1}{}'$; we must prove that $q^c_{R1}{}' \eta \ q^c_{A1}$.

Proof on the form of labels covered by τ. There are three cases.

- $(e'_1, -)\backslash\tau$ or $(e'_1, e'_2)\backslash\tau$. By Definition 4, we have $q^c_{R1} \in Q^c_{R1}$ and $q^c_{R1}{}' \in Q^c_{R1}$. Since $q^c_{R1} \ \eta_f \ q^c_{A1}$ for the whole system, we obtain that there exists q^c_{A1} such that $q^c_{A1} \in Q^c_A$ and $q^c_{R1}{}' \eta \ q^c_{A1}$.
- If $q^c_{A1} \in Q^c_{A1}$, we conclude that $(q^c_{R1}, q^c_{A1}) \in M_1$.
- If $q^c_{A1} \in Q^c_{A2} \setminus Q^c_{A1}$, then $(q^c_{R1}, q^c_{A1}) \notin M_1$ and we are done.

- $(-, e'_2)\backslash\tau$. Impossible because of Definition 4.

Lemma 8 (Old or New Deadlocks Condition)
Assume that $q^c_{R1} \nrightarrow$ *with* $q^c_{R1} \in Q^c_{R1}$*; we must prove that either* $q^c_{A1} \nrightarrow$ *or* $\exists q^c_{A1}{}'.(q^c_{A1} \stackrel{e}{\rightarrow} q^c_{A1}{}'$ *and* $q^c_{R1} \in D_1)$.

- Suppose that q^c_{R1} *is a deadlock for* $SR_1 \|_{Synch'} SR_2$*. As we have* $q^c_{R1}\ \eta_f\ q^c_{A1}$ *for the whole system, we have two cases:*

- *either* $q^c_{A1} \nrightarrow$*;*
  - *If* $q^c_{A1} \in Q^c_{A1}$*, we conclude that* $(q^c_{R1}, q^c_{A1}) \in M_1$.
  - *If* $q^c_{A1} \in Q^c_{A2} \setminus Q^c_{A1}$*, then* $(q^c_{R1}, q^c_{A1}) \notin M_1$ *and we are done.*
- *or* $\exists q^c_{A1}{}'.(q^c_{A1} \stackrel{e}{\rightarrow} q^c_{A1}{}' \wedge q^c_{R1} \in D_{12})$*. Impossible since* $D_{12} = \varnothing$.

- Suppose that q^c_{R1} *is not a deadlock for* $SR_1 \|_{Synch'} SR_2$*. Since* $q^c_{R1} \nrightarrow$ *for* SR_1*, then* $q^c_{R1} \stackrel{e}{\rightarrow} q^c_{R1}{}'$ *for* SR_2*. Then we have* $q^c_{R1} \in Q^c_{R2}$*. We have* $q^c_{R1}\ \eta_f\ q^c_{A1}$ *for the whole system and:*

- *either* $q^c_{A1} \nrightarrow$*;*
  - *If* $q^c_{A1} \in Q^c_{A1}$ *we conclude that* $(q^c_{R1}, q^c_{A1}) \in M_1$.
  - *If* $q^c_{A1} \in Q^c_{A2} \setminus Q^c_{A1}$ *then* $(q^c_{R1}, q^c_{A1}) \notin M_1$ *and we are done.*
- *or* $\exists q^c_{A1}{}'.(q^c_{A1} \stackrel{e}{\rightarrow} q^c_{A1}{}')$*;*
  - *If* $q^c_{A1} \in Q^c_{A1}$*, we verify* $D_1 \setminus Q^c_{R2} = \varnothing$ *because of* $q^c_{R1} \in Q^c_{R2}$ *and we conclude that* $(q^c_{R1}, q^c_{A1}) \in M_1$.
  - *If* $q^c_{A1} \in Q^c_{A2} \setminus Q^c_{A1}$*, then* $(q^c_{R1}, q^c_{A1}) \notin M_1$ *and we are done.*

Lemma 9 (Lack of τ-divergence Condition)
We must prove that there is no $(q^c_{R1} \stackrel{\tau}{\rightarrow} q^c_{R1}{}' \stackrel{\tau}{\rightarrow} q^c_{R1}{}'' \stackrel{\tau}{\rightarrow} \ldots \stackrel{\tau}{\rightarrow} \ldots)$ *in* M_1.

Proof by contradiction. We suppose that $(q^c_{R1} \stackrel{\tau}{\rightarrow} q^c_{R1}{}' \stackrel{\tau}{\rightarrow} q^c_{R1}{}'' \stackrel{\tau}{\rightarrow} \ldots \stackrel{\tau}{\rightarrow} \ldots)$ *exists. Since* $q^c_{R1}, q^c_{R1}{}', q^c_{R1}{}'', \ldots \in Q^c_R$*, we have a* τ*-divergence for the whole system.*

Lemma 10 (Non-determinism Preservation Condition)
Assume that $q^c_{A1} \stackrel{\alpha}{\rightarrow} q^c_{A1}{}'$*; we must show that there exist* $q^c_{R1}{}'$, $q^c_{R1}{}''$ *and* $q^c_{A1}{}''$ *such that* $q^c_{R1}{}' \eta\ q^c_{A1}$, $q^c_{R1}{}' \stackrel{\alpha}{\rightarrow} q^c_{R1}{}''$, $q^c_{A1} \stackrel{\alpha}{\rightarrow} q^c_{A1}{}''$, *and* $q^c_{R1}{}'' \eta\ q^c_{A1}{}''$.

Proof on the form of α*. There are three cases.*

- $\alpha = (e_1, -)$ *or* $\alpha = (e_1, e_2)$*. By Definition 4, we have* $q^c_{A1} \in Q^c_{A1}$ *and* $q^c_{A1}{}' \in Q^c_{A1}$*. Since* $q^c_{R1}\ \eta_f\ q^c_{A1}$ *for the whole system, we obtain that there exist* $q^c_{R1}{}'$, $q^c_{R1}{}''$ *and* $q^c_{A1}{}''$ *such that* $q^c_{R1}{}' \eta\ q^c_{A1}$, $q^c_{R1}{}' \stackrel{\alpha}{\rightarrow} q^c_{R1}{}''$, $q^c_{A1} \stackrel{\alpha}{\rightarrow} q^c_{A1}{}''$ *and* $q^c_{R1}{}'' \eta\ q^c_{A1}{}''$*. By Definition 4, we conclude that* $q^c_{R1}{}'$ *and* $q^c_{R1}{}''$ *belong to* Q^c_{R1}*, and that* $q^c_{A1}{}''$ *belongs to* Q^c_{A1}*. We conclude that* $(q^c_{R1}, q^c_{A1}) \in M_1$.

- $\alpha = (-, e_2)$*. Impossible because of Definition 4.*

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399